Sedit Manual

T.W. Steiner

1995

T.W. Steiner, 2246 E. 6th Ave.
Vancouver, BC, V5N- 1R1, Canada
e-mail tsteiner@creo.bc.ca or steiner@sfu.ca

Chapter 1

Editor

1.1 Introduction

The sedit editor is a fully featured, professional, folding, text and binary editor. The basic command set of the editor is modeled after the E PC editor. This editor is, however, a functional superset of E with many useful additional features (except that there is no REXX support). The OS/2 EPM editor is another editor with the same ancestors. Users of E or EPM will quickly feel at home with this editor since basic operation is quite similar.

1.2 Highlights

Highlights of this editor are as follows:

- Complete GUI application with versions for either OS/2 and W95/WNT with all editor commands available either from drop down menus or by using Ctrl and Alt key combinations. Appearance and operation of this editor are highly configurable.
- Edits text and binary files of arbitrary size and line length.
- Edits binary files in either hex or in a special alphanumeric coded form that allows easy identification of strings in a binary file.
- Unlimited undo of previous changes and corresponding step for step redo. Lines can be restored to their previous state even if subsequent edits on other lines have been made that are to be kept.
- Optional syntax colouring with user configurable colours, comment delimiters and preprocessor delimiters. Colour is used to differentiate plain text, comments, quotes, preprocessor lines, and unbalanced brackets.
- There is full folding support with 6 different fold mechanisms. Four of these are toggles which displays either only the lines starting in the first column, only the changed lines, only lines containing a place mark or only the lines containing the last search string. The remaining two are incremental and selectively hide or display portions of the text. One of these is indentation based and the other requires fold tags embedded in a comment in the file.
- Column editing features to allow movement of columns of text. This allows easy change of indentation for example and is extremely useful for manipulating columns of numbers. Columns can be moved, copied, deleted, overlaid and filled.
- Three types of marking. Lines, blocks and industry standard marking.

- Bracket matching of all sorts of brackets and comment delimiters.
- Single key left justification of moved code blocks with first non-blank line above.
- Run programs such as dir, grep or compilers from the command line. These may optionally run as a separate thread so you can continue to edit while they run or synchronously for predictable behaviour in a macro. Output from programs started from the command line is piped into another file in the ring of loaded files. Files can be loaded from these read only output files by double clicking on any file name. A trailing number, if any, is interpreted as a line number.
- Programming support. Run compilers from the command line and jump to file and line containing the error by double clicking on the file name. In OS/2 version open on-line programming reference information by double clicking on key words.
- Changed lines optionally appear in a different colour from the unchanged text.
- A whole ring of files can be loaded up at once allowing painless copying of text between files. One nice feature is the ability to load up a set of files in a ring and rapidly cycle through them using then Ctrl N and Ctrl P key combinations. Text can be copied between these files by using the block and line mark commands.
- This editor has very few built in hard coded limits. The number of files simultaneously loaded is limited only by available memory. The number of lines in a file or the length of a line is also limited only by the size of available memory or the size of a 4 byte integer whichever is smaller.
- The editor has very powerful search options which allow searching with any of the following modifiers singly or combined. Search up, search marked block only, ignore case, whole word only, loop through all loaded files or interpret as regular expression. For search and replace a dialog allows the following; replace, skip, replace all, quit, undo last replace, and redo last skipped.
- There are all also some features to facilitate carrying out complex repetitive editing tasks. In particular sequences of key strokes can be saved and assigned to a function key for subsequent reuse. Furthermore, any command including function key macros can be repeated a number of times automatically by entering a command multiplier. Bound macros are saved between editing sessions.
- The editor is completely key configurable. A set of keystrokes can be mapped to an editor function simply by specifying the sequence in the auxiliary key map file. The combination of key mapping and key macros allow the editor to imitate the user interface of other editors or allow the construction of a custom user interface to the taste of the user. If key mapping is on then the accelerators in the drop down menus are updated to show the mapping on the fly.
- Help on any menu item may be obtained by right rather than left clicking on a menu choice. This will open up a help dialog with information on the given command. The help text is automatically updated to reflect the current key mappings if any.

1.3 Control keys

Basic operation of the editor is largely self explanatory. Text is input by straight typing anywhere on the screen. Cursor movement is controlled by the arrow keys and auxiliary keypad keys as per their standard definitions as well as accelerated movement with Ctrl combinations. The cursor can also be positioned using the mouse by clicking with the left button at the desired cursor location. The cursor position control keys are as follows:

Un-mapped Cursor Motion Keys

left arrow	left one space
right arrow	right one space
up arrow	up one line
down arrow	down one line
end	end of line
home	beginning of line
PgDn	scroll down one screen
PgUp	scroll up one screen
Center Key	fast motion toggle
Ctrl left	left one word
Ctrl right	right one word
Ctrl up	scroll up five lines
Ctrl down	scroll down five lines
Ctrl end	end of file
Ctrl PgDn	end of file
Ctrl home	beginning of file
Ctrl PgUp	beginning of file
Alt home	top of page
Alt end	bottom of page
Alt PgUp	scroll up half a page
Alt PgDn	scroll down half a page

Un-mapped Editor Control Keys not discussed in other sections

Ctrl a	set place mark
Ctrl A	save all changed files
Ctrl b	bracket match {, (, [, <, or /*
Ctrl c	change case of character under cursor
Ctrl C	opens configuration notebook (OS/2)
Ctrl d	delete character
Ctrl D	read in the displayed macro list
DEL	delete character
Ctrl e	delete to end of line
Ctrl h	destructive backspace
Ctrl H	open select other loaded file dialog
Ctrl g	execute file (C interpreter version)
Ctrl i	tab
Ctrl j	join next line to current line
Ctrl k	cut current line at cursor
Ctrl K	list the currently defined key mappings
Ctrl l	return to previous set place mark
Ctrl L	list the currently defined macros
Ctrl m	insert new line (also return key)
Ctrl M	open command line input dialog

Ctrl n	edit next file in ring
Ctrl o	switch to other window if screen split
Ctrl O	open file select dialog
Ctrl p	edit previous file in ring
Ctrl q	quit without saving file
Ctrl r	redo previous undo
Ctrl s	swap current and next character
Ctrl t	toggle insert/replace
Ctrl u	undo changes to last modified line
Ctrl w	write file to disk
Ctrl W	write file to disk with different name
Ctrl x	delete line
Ctrl y	split edit window horizontally in two
Ctrl Y	read in the displayed key mappings
Ctrl BACK	delete line
Ctrl z	zap word
Ctrl DEL	zap word
Alt a	alternate binary representation
Alt B	set breakpoint (interpreter version)
Alt e	execute last macro
Alt D	defines (interpreter version)
Alt G	global vars (interpreter version)
Alt h	view only lines with place mark
Alt i	clear all place marks
Alt I	step into (interpreter version)
Alt k	enter next char literally
Alt L	local vars (interpreter version)
Alt O	step over (interpreter version)
Alt p	position curs at previous location
Alt q	restore a line from undo record
Alt r	re-flow paragraph, stops at blank line
Alt O	reset execution (interpreter version)
Alt s	view lines containing search string
Alt t	teach (end) new macro
Alt v	view only changed lines
Alt x	enter control character
Alt z	zero command multiplier
Alt 0-9	enter command multiplier

The following section provides more detailed information for a selection of the above commands that are perhaps not inherently obvious.

- The Ctrl a command sets a place mark on the current line in the current file. Any number of place marks can be set. Hitting Ctrl l will return the cursor to these place marks in the reverse order that they where set. To remove a place mark hit Ctrl a again on the line with the place mark and it will be removed. Alt i removes all currently set place marks and Alt h shows only lines on which a place mark has been set.
- The bracket match command Ctrl b moves the cursor to the bracket matching the one under the cursor. If the cursor does not move it means that a matching bracket could not be found or that the cursor is not presently sitting on a bracket. The matchable characters are <, >, (,), {, }, [,

] and the comment delimiter pairs. The comment delimiter pairs are set in the syntax colouring configuration dialog and are /* and */ by default.

- Ctrl c changes the case of the letter under the cursor. If the case of a whole region is to be changed. Block mark the region (Alt b) and then use the change case option of the block fill command (Alt f).

- Ctrl j, k, m. Unlike most editors hitting return (Ctrl m) does not split the current line at the cursor. Instead a line is split at the cursor using Ctrl k. This usage may take some getting use to but it is the same as used by the editor E and optionally by the EPM editor. If, however, you prefer more conventional return key behaviour this can be set in the configuration menu. Ctrl j is used to join two lines together. Return or Ctrl m inserts a new blank line below the current line but the current line is not split.

- Ctrl n, p. If more than one file is currently in memory then Ctrl n switches to the next one in the ring while Ctrl p switches to the previous file. If the switched to file has been modified externally to the editor since it was last seen you will be prompted for an optional reload.

- Ctrl q. Quits the current file which will be removed from memory and all changes will be lost. If changes have been made and the file has not been saved then you will be prompted for confirmation. It is possible to configure the editor to prompt for a confirmation before over writing an old version of the file.

- Ctrl u, r. Undo the change to the last edited line. The number of undo levels is set in the configuration dialog and is essentially unlimited. By repeatedly hitting Ctrl u all the changes made to a file since it was loaded can be undone. Ctrl r re-does a previous undo. The changes to a file can only be undone in the order that they where made except that any line for which a line undo record exists can be restored using Alt q. Only simple edits that only operate on a single line generate undo records of the type that can be undone using Alt q. In some instances it may be desirable to turn off the undo record generation for example if column manipulations are to be done on very long columns since in that case the entire file will have to be duplicated may times for the undo records which will adversely affect performance. Undo record generation may be turned off by setting the undo limit to zero in the config dialog of the file menu.

- Ctrl w. Saves the current file to disk using the file name displayed on the command line. The file name can be changed by moving to the command line ESC and using the rename command. The file can also be saved with a different name using the “save as” menu item of the file menu. Before writting the existing file of this name is renamed “editfile.bak”. Thus writting a file may be undone provided no subsequent files were written.

- Ctrl y splits the editor window horizontally in to two windows separated by a second command line. Hitting Ctrl y again un-splits the screen. Use Ctrl o or the left mouse button to change the current window. These commands are not available in the version of the editor bundled with Splot.

- Alt a selects the alternate binary representation. This key is only active for files loaded using the binary switch (-b). The default binary representation is hex with two hex characters per byte and 32 bytes per line. The alternate representation uses the following codes: Printable characters appear with a leading underscore and control characters with a leading $\wedge$. This second representation also has 32 bytes per line and is more useful for identifying text in binary files. Individual hex numbers may be converted to decimal or the reverse by using the command line commands “hex”, “dec” and “asc”.

- Alt e executes the last defined macro see section 1.7.

- Alt i removes all previously set (Ctrl a) place marks.

- Alt k interprets the next character as a literal without re-mapping. This key may be useful for enter characters that have been mapped onto editor commands. This should not be needed with the default configuration since only Ctrl and Alt keys are used to control the editor.
- The Alt p command is useful if the cursor was accidentally moved away from the region of interest with a Ctrl Home or search for example. In these situations Alt p will restore the cursor to its previous location. This action is similar to using place marks except that unlike place marks these are automatically set by any command that causes the cursor to jump by more than one line. Only one previous location is remembered though.
- The Alt r command re-flows a paragraph with the right and left margins and the first line indentation set as specified in the config dialog. It starts at the current line and stops at the first encountered blank line. If a region is Alt b marked and the cursor is currently in the marked region the marked region only will be reformatted with the right and left margins set by the marking and no initial indentation. If syntax colouring is on then the file is assumed to be program code and is reformatted as code rather than text using a set of rules that are currently not configurable. In this case the region to be reformatted must be line marked.
- Alt t starts and ends the definition of a new macro see section 1.7.
- Alt x queries the user for the 3 digit hex code of the character to be inserted. Alt x can thus be used to enter characters for which there is no keyboard key such as the characters above 127 in the ASCII table.
- Alt 0-9, z. These keys are used to enter a command multiplier to be used on the next command. Entering a multiplier has the same effect as entering the following keystroke n times. Alt z resets the multiplier to zero.
- Alt F4. The system default close application key removes all loaded files and terminates the editor. If there are any changed unsaved files a confirmation dialog appears. Other system menu Alt F# keys perform their usual task. These keys are intercepted before reaching the editor and thus cannot be used for bound macros.

1.4 Searching

The editor has very powerful search options which allow searching with any of the following modifiers singly or combined. Search up, search marked block only, ignore case, whole word only, loop through all loaded files or interpret as regular expression. For search and replace a dialog allows the following; replace, skip, replace all, quit, undo last replace, and redo last skipped. The initial search is done by opening the search dialog from the menu or with Ctrl S (Ctrl R for search and replace). A search can also be initiated from the command line by prefixing the search string with a '/' or any other punctuation not used for other purposes. A search and replace from the command line is of the form "c/str/rep/[-cblrw]". The square brackets denote the optional search options if any and are not themselves to be entered. Subsequently, the same string may be searched for using Ctrl f. The editor keys used for searching are as follows:

Un-mapped Search Control Keys

Ctrl f	find next occurrence of search string
Ctrl F	change dir and find search string
Ctrl R	opens the search and replace dialog
Ctrl S	opens the search dialog
Alt *	find next word like current word

Alt #	find prev word like current word

- Ctrl f finds the next occurrence of the search string. The first occurrence must be found either by going to the command line with ESC and entering the desired search string after a '/' as in "/string" or else using the search dialog from the menu bar.
- Ctrl F. Changes the current search direction and locates the next search string. Thus if the current direction is down then a Ctrl F finds previous location of the search string having set the direction to up. If it is now desired to continue searching in the up direction use Ctrl f.
- Alt * and Alt # finds the next or previous word matching the word currently under the cursor respectively. This is equivalent to doing a search for the current word with the /w whole word flag turned on so only whole words matching the current word are found. The current search string is not updated however.

The search options are specified using the buttons in the search dialog or with the optional command line switches. For example: ""-w" searches up from current cursor position and looks only for "string" that is not part of a larger word. The optional search suffixes are:

- search up (down is default)

b search marked block only

c ignore case

l loop through all files

r interpret as regular expression

w match whole words only

Of these the most complex and most powerful is the interpret as regular expression option. Regular expressions allow search strings to be built that will match complex criteria. A simple example is "[0-9]+\.[0-9]+" which will match any fixed point number. For regular expressions certain characters have special meaning:

∧ start of line or not the following characters in a set

$ end of line

. any character

\ next character is literal. Used to override special meaning of next character.

* match previous character or group zero or more times

+ match previous character or group one or more times

[start of a set.

] end of a set.

- range delimiter

(start of a sub-expression

) end of a sub-expression

The following are some simple examples:

[a-zA-Z]+	match any word containing only letters
[aei0-9]	match a,e,i and 0 through 9
[∧aei0-9]	match anything but a,e,i and 0 through 9
a(ab)*b	matches ab aabb aababb aabababb etc.

If a "grep" (A UNIX utility program which searches files for a regular expression) utility is run from the command line the grep search string automatically becomes the editor search string. If your grep does not have the name grep set up a batch file or rename it to activate this feature.

1.5 Folding

Folding refers to the ability of the editor to selectively hide parts of the file in order to give a better overview of the whole file. This may be used to rapidly find sections in the text. The folding control keys are as follows:

Folding Keys

Ctrl v	first column lines only
Alt v	changed lines only
Alt s	lines containing search string only
Alt h	lines containing place mark only
Alt -	Fold to next lower fold tag
Alt +	Unfold to next greater fold tag
Ctrl -	Fold to next tab position to left
Ctrl +	Unfold to next tab position to right

The first 4 of these commands are toggles i.e. hitting Ctrl v twice returns the file to the unfolded state. Only one type of folding may be active at any one time. The file may be fully edited while folded but marked blocks can not be transferred across folds. These commands are useful for rapidly finding a section of the file.

Ctrl v causes only lines starting in the first column to be displayed. If the text is organized so that only section headings or sub-routine headings start in the first column then Ctrl v can be used to rapidly locate a section in a file. To do this hit Ctrl v then move cursor to desired sub-routine heading and hit Ctrl v to expand again. Alt v and Alt s work similarly except they show only changed lines and lines containing the last search string respectively. For example in a large case statement the start of any branch may be found by doing a search for "case" and then selecting Alt s. This will show only lines containing the word "case" on the screen. The cursor may then be put on the desired line and the file expanded again. Alt h can be used to view only lines on which a place mark has been previously set using Ctrl a. When folded the visible lines may still be edited.

The Alt - and Alt + pair are complementary keys and unlike the toggles above several levels of this type of folding are possible. To use this folding mechanism fold tags must be inserted in the text. By default these are of the form // Lnnn where nnn is an integer. This is in the form of a C++ comment and can thus be put into source code without perturbing the program. The form of the tag string may be edited in the config dialog by changing the fold tag regular expression. Note that the fold tags must contain an integer counter as part of the tag. The first time Alt - is hit the largest value of all the found fold tags is found and the lines between tags of this number and tags with smaller numbers are hidden. A subsequent Alt - will decrement the folding tag number and hide more lines. Thus multiple levels of folding are possible with this mechanism. An Alt + will increment the fold tag number and reveal the corresponding hidden lines. A sufficient number of Alt + hits will return the file to its unfolded state. This folding mechanism differs from the others since the file must be modified to contain the fold tags as special comments but it has the flexibility to place fold markers anywhere in the file. This type of folding is not available in the version of the editor bundled with Splot.

The Ctrl - and Ctrl + pair is similar in operation to Alt - Alt + but does not use fold tags. Instead on the first Ctrl - the greatest number of tab indentations (depends on the set tab size) on the current page is determined and the file folded to hide all the lines indented this far. A further Ctrl - decrements this number and hides more lines. Ctrl + counter acts a Ctrl -. This type of folding is useful with structured program code that uses a tab indentation to delineate a block. Thus for example the start of both branches of an if else block may be seen by hiding the bodies. This type of folding is not available in the version of the editor bundled with Splot.

1.6 Block manipulations

Block mark and line mark are somewhat different. Use line mark for moving, copying or deleting one or more whole lines and block mark for moving, copying, deleting, filling or overlaying columns of text. Note also that a block marked region can also be moved to the left with Ctrl ∧ and to the right with Ctrl _ but the behaviour is different than if the Alt arrow keys are used. Alt left and Alt right (or Ctrl _ and Ctrl ∧) are meaningless for line marked regions. Block marked regions can furthermore be filled with a character, a string or incrementing or decrementing numbers. Marked blocks are also used to constrain a search and indicate the desired margins for re-flowing a paragraph. Alt n marking is the industry standard marking useful for text but not of much use for programming. Regions can also be marked by dragging the mouse with one of the three buttons down. The type of region mark created depends on the settings in the "config" notebook or menu. Lastly, Alt n Type marking can be achieved by moving the cursor with the shift key held down. Alt b marked regions may be filled with a single character using the Alt f command. This command can also be used to change the case of all letters in a block. Alt f can also be used to fill with incrementing or decrementing numbers starting with the numerical value at the top of the marked block. To fill an Alt b marked region with a word use the "fill" command from the command line. In order to fill a B marked block with a string be sure that the width of the block in characters matches the width of the fill string otherwise the fill word will wrap around to the next line or repeat.

Block Manipulation Keys

Alt b	mark column block start, end
Alt c	copy marked block
Alt d	delete marked block
Alt f	fill B marked block with char, change case or inc/dec
Alt j	justify last copied block to match lines above
Alt l	mark LINE(s) start, end
Alt m	move marked block
Alt n	standard marking start, end
Alt o	overlay B marked block
Alt u	un mark block
Alt w	write block to 'editblck.tmp'
Alt y	yank back deleted block
Ctrl_	shift B marked block right
Ctrl∧	shift B marked block left

1.7 Macros

There is also a macro capability for repetitive, complex editing tasks. The editor is taught a macro by hitting the Alt t combination. After this all subsequent keystrokes are stored as the macro definition until Alt t is hit again. There is a 128 keystroke limit on the length of a macro. A macro can consist of any keystrokes except Alt t and Alt e. At the end of the macro definition the macro can be bound to a function key by hitting the desired function key. Available function keys are F1 - F12 as well as Shift, Alt and Ctrl F1 - F12. In OS/2 and W95/WNT some of these are not available as they are used by the system to resize windows etc. and are trapped before they even reach the editor. As a special case F10 has been partially activated by defining it as an accelerator key in the resources. As a consequence a macro defined for F10 in the editor.cfg can be executed but no macro can be bound to it from within the editor using Alt t. A bound macro is subsequently executed by hitting that function key. The predefined macro for that key will be lost. The last learned macro, whether bound or unbound, can be executed by hitting Alt e. If the macro involves a search and another item is not found the rest of the macro is not executed. If macros have been bound to function keys they will be written to the current configuration file editor.cfg which so that the macros will be preserved in between editing sessions. The editor.cfg file as received contains a number of predefined macros that emulate the E editor usage of the function keys. For example F2 is predefined as save (Ctrl w), F3 is predefined as quit (Ctrl q) and F4 is save and quit (Ctrl w Ctrl q). These may of course be over written. The name of the configuration file is different for versions of the editor bundled with the C interpreter and splot in which case the names are cterp.cfg and splot.cfg respectively. The currently defined macros may also be viewed and edited in plain english by using the "Macro list" (Ctrl L) command.

A repeat factor for a keystroke can also be entered by prefixing the command with a multiplier which is input using the Alt number keys. The multiplier can be reset to zero during input using Alt z. This feature also facilitates repetitive editing as an operation can be repeated several times. This is particularly powerful in conjunction with macros.

1.8 Command Line

Lastly, there are a number of commands which require typed input. Hitting Esc places the cursor at the beginning of the command line which is on the last row and ordinarily displays some status information. Alternatively a text entry dialog can be opened using Ctrl M or the "Cmd" menu item. Once on the command line text can be input for the following functions:

Command Line Operations (requires text input so terminate line with a return)	
Esc	goto or leave command line
	Change current line
number	goto line number n
+number	down n lines
-number	up n lines
Change current column offset	
@number	start at column n

@+number	scroll left n columns
@-number	scroll right n columns

Search and Replace.
find string str (/ actually any punct)
/str[/-bclrw] return
replace str1 with str2
c/str1/str2[/-bclrw] return

The optional search suffixes are:

/-	search up (down is default)
/b	search marked block only
/c	ignore case
/l	loop through all files
/r	interpret as regular expression
/w	match whole words only

Add other files to ring
e name1,name2,...[-r]
The optional edit suffixes are

-r	read only
-b	binary mode

display directory and choose one to edit
    e [*.*]
insert named file at current location
    m name or merge name
rename current file to name
    r name or rename name
show and modify editor configuration
    cfg or configure
exit without saving changes
    quit
fill marked block with string
    fill string
sort lines using field number col as key
    sort col
find differences between current and named file
    d name or diff name
change current drive and or path
    cd y:\path
convert decimal num to hex
    hex num
convert hex num to decimal
    dec num
convert number to ASCII
    asc num
previous command line entered
    up arrow
next command line entered
    down arrow
send command to operating system

os command
also any unrecognized string

There is an important difference between OS system calls initiated using the “os” prefix on the command line and those initiated without the “os” prefix. The later are run as a separate thread which allows editing to continue while the called program runs. The system calls initiated with the “os” prefix however do not run as a separate thread. This means that the editor will wait until the system call is done before allowing any further editing. This behaviour is desirable when a system call is part of a macro since then the macro can be made to wait for the results of the system call.

1.9 Configuration Files

The startup file editor.cfg preserves the values selected from the configuration menu item between sessions. You can have multiple editor.cfg files in different directories since the current directory plus parent directories two levels up are searched before using the default editor.cfg stored in the same directory as the executable. This is useful since a configuration suitable for text (auto wrap on, auto indent off) is not ideal for programming (auto wrap off, auto indent on) thus text specific editor.cfg files may be kept in text directories and programming specific ones in program directories. The editor.cfg file is plain ASCII and can thus be edited with a text editor as an alternative means of changing the settings or the macro definitions. If changes have been made to the configuration the editor will update the editor.cfg in the directory from which it was read. Thus in order to create a custom cfg file for use only within one directory copy the default editor.cfg to the current directory before starting the editor.

The current macro definitions bound to function keys are also stored in the editor.cfg file but in an encoded form. Thus, it is easier to view and modify macros from within the running editor by selecting the list macros command in the macro menu. There the macro codes will be translated to plain English. Note that the macro definitions must be in terms of the un-mapped keystrokes.

The key mappings in use if any are stored in the file editkey.map which has the key names in plain English. It may also be viewed or edited from within the editor by selecting the keymap menu item in the Config menu.